A Machine-Learning Approach for Communication Prediction of Large-Scale Applications

Nikela Papadopoulou, Georgios Goumas, Nectarios Koziris
National Technical University of Athens
School of Electrical and Computer Engineering
Email: {nikela, goumas, nkoziris}@cslab.ece.ntua.gr

Abstract

In this paper we present a machine-learning approach to predict the total communication time of parallel applications. Communication time is heavily dependent on a very wide set of parameters relevant to the architecture, runtime configuration and application communication profile. We focus our study on parameters that can be easily extracted from the application and the process mapping ahead of execution. To this direction we define a small set of descriptive metrics and build a simple benchmark that can sweep over the parameter space in a straightforward way. We use this benchmarking data to train a robust multiple variable regression model which serves as our communication predictor. Our experimental results show notable accuracy in predicting the communication time of two indicative application kernels on a supercomputer utilizing from a few dozen to a few thousands processing cores.

Keywords

performance prediction; communication time; MPI applications; large-scale systems;

I. Introduction

As supercomputers grow in cores and computational capacity, the challenge of delivering high levels of performance at extreme scales is becoming harder. Exascale systems are expected to have hundreds of thousands of nodes with hundreds of cores each, yet for large classes of parallel applications the cost of communication often rises to a degree high enough to impede scalability. Execution time and in particular communication time is affected by application characteristics and problem size, the system architecture and the runtime configuration, which clearly constitute a complex set of parameters, hindering any intuitive assumptions about performance at scale.

In principle, the ability to predict the performance of parallel applications enhances decision making at many levels: supercomputer users can predict the scalability of their applications, schedulers can make optimal decisions with respect to power and performance, auto-tuning frameworks can decide to enable or disable code optimizations at runtime. However, the task of devising performance models is intricate. Particularly for communication at large scale, the number and complexity of factors that can affect performance is dramatically high, as in most real-world applications, communication takes place as a burst of message transmissions at a certain phase of the execution, leading to complicated resource sharing effects. Such effects have been studied through benchmarking in previous works [1,2], yet still lack formal definitions. Current state-of-art models for communication, such as the classic latency-bandwidth model and the Log(G)P [3] family of models model network characteristics, however they provide only a local view of communication and overlook the nature of most real-world parallel applications, where multiple processes communicate concurrently. A more holistic view of communication performance is presented in the work of Jain et al. [4] and Bhatele et al. [5], who utilize the network performance counters of BlueGene/Q to build performance models for the communication of real-world applications, using supervised learning techniques. Their work delivers significant prediction accuracy, however, the modeling approach is unsuited for forecasting the communication time ahead of execution.

In this paper, we propose a machine-learning approach to communication time prediction for point-to-point communication of MPI applications. In particular, we utilize a simple benchmark to extract information on the interaction of the interconnect architecture and the communication time and perform statistical analysis to identify possible predictors and their relation to observed communication times.

We construct a model for communication time with selected predictors and compute its coefficients with robust multiple variable regression, utilizing measurements from our benchmark. Obscure network effects are hidden within the coefficients. We derive our predictors from the application characteristics and its mapping, all available at runtime ahead of execution. The philosophy of our approach is not attached to a particular system or architecture and can be rather easily reproduced for other systems by adapting or extending the set of predictors. To our knowledge, this is the first work that achieves communication prediction of a communication phase of an application, prior to execution, without the need to model and combine communication primitives or network effects. We evaluate our model in predicting the communication time of two application kernels for various problem sizes and execution configurations.

The rest of the paper is organized as follows: Section II presents some background information on our problem definition and platform. Section III presents the machine learning methodology. Section IV presents the experimental evaluation and Section V concludes the paper and presents future work directions.

II. Background

A. Problem definition

In this work, we focus our study on MPI applications with iterative phases of computation and communication as is typical in numerous real-life scientific applications. We deal with non-blocking, point-to-point communication, assuming that the computation is rather balanced and that computation and communication are two discrete phases. In addition, we examine applications where all MPI processes have the same number of messages and constant message sizes.

B. The Vilje Supercomputer

Our target platform is *Vilje*, a supercomputer installation at NTNU, the Norwegian University of Science and Technology, ranked 127-th in the recent Top500 list. Vilje is an SGI system that consists of 1404 Intel Xeon-E5 dual eight-core nodes on 19.5 racks, interconnected with Infiniband FDR. The machine topology is an enhanced hypercube, where redundant links are added to available switch ports at the lower dimensions of the hypercube. Each switch connects to 18 nodes. The MPI library built in the system is a component of the SGI Message Passing Toolkit.

The process of formulating a communication performance model comes with several challenges, as Vilje is a system of high utilization on which we operate with the privileges and constraints of an average user. First, the benchmarking process is a set of probes on a full system. Second, as the configuration of the PBS scheduler does not allow for specific node selection at the time, we have no control over the shape of the allocation given upon a request for a job execution. This fact, along with the high utilization degree of the system, has a substantial impact on the process of benchmarking. Nodes of a requested allocation are unevenly distributed over multiple switches and racks. The outcome of these irregular node allocations is a high variance of the communication time between executions of the same application on different requests for allocations. For many configurations, the percent differences exceed 100%. The possible interference from other jobs that utilize shared network components also hinder the modeling.

III. Communication Modeling

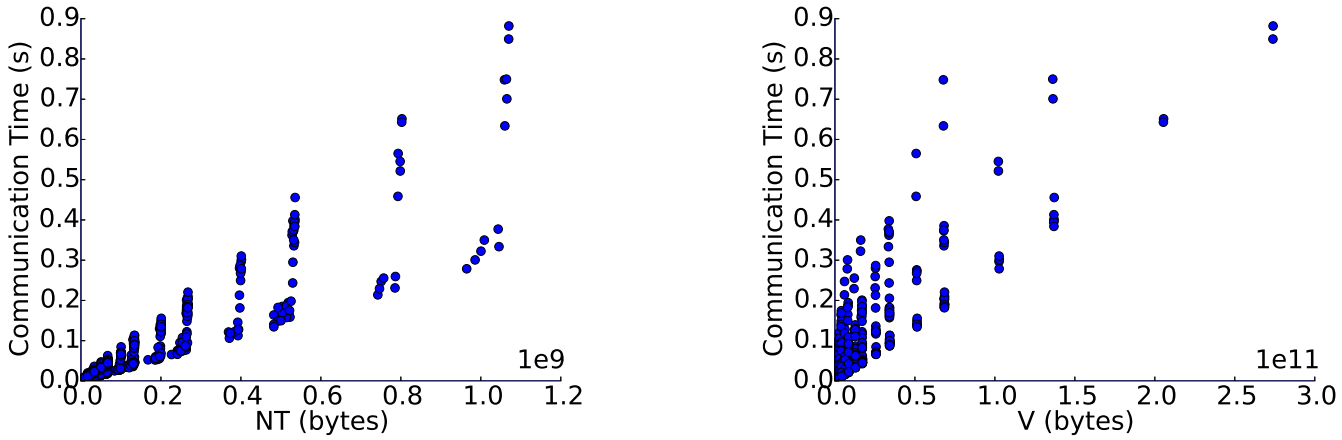
The parameter space for communication performance is very large. Initially, we need to clearly enumerate and quantify the communication-descriptive metrics, derived from the application communication profile and the process mapping. We investigate parameters that can be extracted at compile time or runtime, just after the actual allocation of processes and before the actual execution of the application. Then, we employ a benchmarking step to relate communication time to descriptive metrics and generate a training set for our model. This benchmarking step needs to be applied only once (e.g. after the initial deployment of the execution platform) and thus its execution overhead is not within the critical path. In

the next step, we need to decide upon an effective machine learning approach, perform variable selection and select the model form, making use of statistical methods. Finally, we compute the model coefficients (in our case with robust multiple variable regression). The specifics of this methodology are presented in the following paragraphs.

A. Descriptive Metrics

We base our approach on metrics that are known at runtime and can be extracted prior to execution. We opt to base on quantifiable metrics from the application communication profile and the basics of process mapping onto the underlying system. Thus, we consider the following metrics:

- *Message Size (in bytes)* (S): The latency and bandwidth can vary and the software layer protocol may change for different message sizes.
- *Number of Messages per Process* (M): Multiple messages sent from the same process may be serialized or overlapped.
- *Processes per Node* (PPN): A high number of processes per node implies better resource utilization but may also be responsible for delays, as local network components need to serve multiple demands coming from different processes.
- *Nodes* (N): The number of nodes defines the size of the allocation and can relate to many parameters, such as the dilation and the path diversity.
- *Ratio of Internode Communication* (I): The ratio of the total number of messages exchanged between processes on different nodes to the total number of messages exchanged.
- *Process Traffic (in bytes)* ($PT = S * M$): Process traffic can capture local contention effects.
- *Node Traffic (in bytes)* ($NT = PT * PPN * I$): The amount of data injected to the network by the processes hosted on a compute node. This metric is highly correlated to communication time and is capable of identifying congestion on injection links and contention on terminal switches.
- *Total Communication Volume (in bytes)* ($V = NT * N$): The application data flowing through the network during a communication phase. It may reveal network capacity.
- *Messages Injected per Node* ($NI = M * I * PPN$): A different metric for node traffic, in number of messages rather than in bytes. It encapsulates potential overlapping or serialization of messages during their injection from the node to the network.
- *Total Injected Messages* ($VI = NI * N$): As previously, an alternative metric for total communication volume.
- *Number of Switches in Allocation* (SW): A metric for the networking components available for the application's execution. Switches are known hot-spots of contention.
- *Number of Racks in Allocation* (R): An indicator of dilation, as on the hypercube topology of Vilje, nodes residing on different racks are at least at a hop's distance.



(a) (b)

Fig. 1. Correlation plots for metrics with high correlation with communication time: NT and V

- *Estimated Switches per Racks* ($SWR = SW/R$): A combined metric for allocation density. High values of SWR imply a dense allocation, therefore the available links are fewer and will be more stressed. Low values of SWR imply a sparse allocation, where traffic is spread on more links.

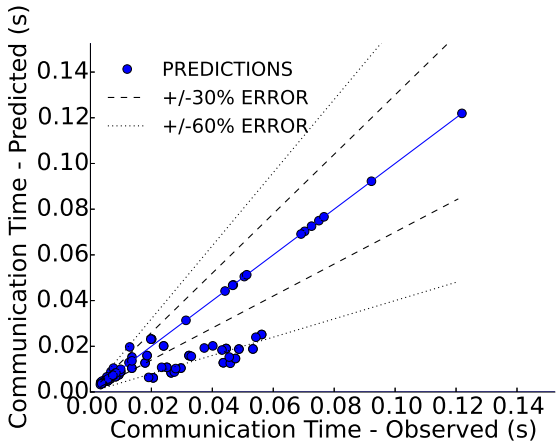
Metrics S, M and PT are extracted from the application communication profile, PPN and N are provided by the user, SW, R and SWR are extracted from the process mapping, while the remaining ones, I, NT, V, NI and VI combine knowledge of the application communication profile and the process mapping. The values of all metrics in study can be extracted ahead of execution.

B. Benchmarking

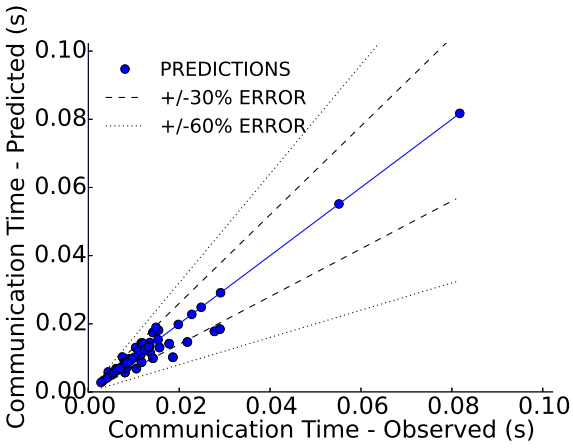
Benchmarking in our methodology serves two purposes: a) to shed light on the impact of the metrics on communication time and b) to extract an incisive dataset to train the model coefficients with sufficient accuracy. We devised our benchmark to resemble a generic application communication phase, where data streams from multiple processes flow concurrently throughout the network, also sweeping the space of parameters formed by the metrics in our study. In our benchmark each process performs simultaneous non-blocking ping-pongs with one or more randomly selected processes, with messages of equal size. The communication times reported from the benchmark are computed as follows: the core ping-pong operations are executed for a few hundred iterations, with an MPI barrier between each, to synchronize the processes and allow for more accurate time measurements. To avoid extreme outliers from OS noise, each process disposes 25% of the iterations with the highest reported communication times, so the maximum time for each process is the third quantile of all iterations. The reported communication time from the benchmark is the maximum time of all processes.

C. Model Building

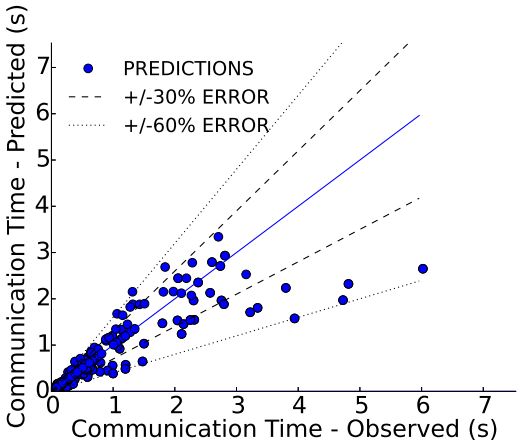
The aim of our performance model is to associate communication time with the set of metrics described before serving as explanatory variables. The task falls into the category of supervised learning, where the training set has already been extracted from benchmarking. To collect the training set, we executed our benchmark on Vilje for various allocations of 8 up to 128 nodes, 1 up to 16 processes per node, 1 to 4 messages per process (accordingly 1 to 4 random communication partners per process) and message sizes of 128 bytes up to 16MB. The benchmark was executed twice, to collect different values for the number of switches and racks, resulting in a training set of 4320 measurements for communication time. Variable selection is an important part of the modeling process. Note that, not all variables discussed in previous



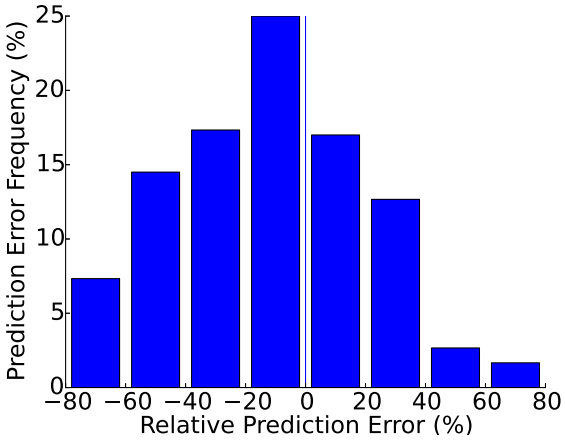
(a) Correlation plots for Area Ia



(b) Correlation plots for Area Ib



(c) Correlation plots for Area II



(d) Relative error distribution for all areas

Fig. 2. Prediction results for 3D-Jacobi and 3D-Advection

sections necessarily influence communication time at a critical extent, on a given execution platform. We applied Pearson's correlation coefficients to decide which metrics have a high impact on communication time. We observed a very high correlation for node traffic and total communication volume and relatively high correlations for message size and process traffic. To inspect whether the correlation is linear and whether there is need for more than one models for different ranges of the critical metrics, we inspected their scatterplots, depicted in Fig. 1, which indeed revealed a linear dependence. However, a scatterplot for node traffic in logarithmic scale showed that there is a different slope for small and large values of node traffic, thus we divided the training set in two areas, named Area I ($NT \leq 128kB$) and Area II ($NT > 128kB$). For Area I, a scatterplot for message size in logarithmic scale revealed again two different areas for different message sizes, Area Ia ($S \leq 4kB$) and Ib ($S > 4kB$). We studied and modeled these three areas separately. The scatterplots also exposed the existence of interactions between metrics, thus we performed clustering against other metrics, to select the interaction terms of the models. Then, we constructed the final model forms by incorporating the selected variables and interaction terms. Finally, we applied robust multiple variable regression on the benchmark data to calculate the model coefficients, using the *Linear Model* class available in Matlab R2013a. We selected robust regression to deal with heteroskedastic errors and influential outliers.

IV. Evaluation

To evaluate the prediction accuracy of our model, we experimented with two parallel application kernels that follow the application model described in Section II. The two kernels are a 3D-Jacobi solver and a 3D-advection solver. For both applications, the processes are arranged in a virtual 3D-cartesian grid (p_x,p_y,p_z) and the original 3D domain (N_x,N_y,N_z) is decomposed into smaller 3D subdomains (N_x/p_x,N_y/p_y,N_z/p_z). The 3D-Jacobi solver exposes a 3D-halo communication pattern, while th 3D-advection solver exposes a 3D-wavefront communication pattern. We predict the communication time for the two application kernels, for various problem sizes, core counts and execution configurations, from 16 up to 8192 processes. 3D-Jacobi has been executed three times, to ensure that our prediction models adequately capture the variations in communication time due to varying process mapping.

Fig. 2 presents a comparison of the observed and predicted communication times. Overall, as demonstrated in Fig. 2(d), our models predict the communication time of the application kernels in study with errors ranging from -80% to 80%, while the average relative error, in absolute values, is 29% and 60% of the predictions lie within a $\pm 30\%$ error margin. Area Ia (Fig. 2(a)) contains a small set of mispredicted points lying around the -60% error line, which correspond to configurations with high core counts, exceeding the core counts in our training set. For Area Ib (Fig. 2(b)), prediction errors are very small, as this area corresponds to large messages and low node traffic. Communication configurations with large message sizes are more predictable, since the communication of large messages depends more on the available bandwidth at different components of the network and is less susceptible to system-level effects and events, while the low node traffic makes the network less prone to congestion. Area II concentrates the majority of our test cases and the corresponding model shows a good prediction ability (Fig. 2(c)). As this is an area where many congestion and contention effects may arise on networking components, we attribute any high prediction errors to the lack of fine-grained process mapping metrics, which would help the modeling of such effects. However, the irregular node allocations on Vilje impedes the precise extraction of such metrics.

V. Conclusions

In this paper, we worked towards the construction of a prediction model for the communication phases of MPI applications with iterative kernels and point-to-point communication patterns. We defined a set of descriptive metrics for communication time, which can be extracted from the application's communication profile and process mapping, ahead of execution. We developed a simple benchmark to explore the relation of the defined metrics with communication time and utilized the benchmark's results to build and train a robust multiple variable regression model for communication performance on Vilje. Our modeling approach can be easily applied to other systems with appropriate adjustment of metrics to the underlying architecture. Prediction of communication time with our performance model for two application kernels, 3D-Jacobi and 3D-Advection, are accurate for a large set of the tested configurations.

Testing our methodology on more interconnection networks of different architectures and topologies and experimenting with more automated machine-learning methods is a work in progress. We also aim to extend the application model in study, to handle irregular communication patterns and collective communication.

Acknowledgments

Nikela Papadopoulou has received funding from IKY fellowships of excellence for postgraduate studies in Greece-SIEMENS program. Georgios Goumas and Nectarios Koziris were supported by project I-PARTS:Integrating Parallel Run-Time Systems for Efficient Resource Allocation in Multicore Systems (code 2504) of Action ARISTEIA, co-financed by the European Union (European Social Fund) and Hellenic national funds through the Operational Program Education and Lifelong Learning (NSRF 2007-2013). This research was supported in part with computational resources at the Norwegian University of Science and Technology (NTNU) provided by NOTUR (http://www.notur.no).

REFERENCES

[1] A. Bhatelé and L. V. Kalé, "Quantifying network contention on large parallel machines," *Parallel Processing Letters*, vol. 19, no. 04, pp. 553–572, 2009.

[2] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: performance degradation due to nearby jobs," in *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 41.

[3] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation," in *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*. ACM, 1995, pp. 95–105.

[4] N. Jain, A. Bhatele, M. P. Robson, T. Gamblin, and L. V. Kale, "Predicting application performance using supervised learning on communication features," in *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 95.

[5] A. Bhatele, A. R. Titus, J. J. Thiagarajan, N. Jain, T. Gamblin, P.-T. Bremer, M. Schulz, and L. V. Kale, "Identifying the culprits behind network congestion," in *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (to appear)*, ser. IPDPS '15. IEEE Computer Society, May 2015.